Parallel Sparse Cholesky Factorization with Spectral Nested Dissection Ordering

Alex Pothen* Edward Rothberg† Horst Simon‡ Lie Wang §

Abstract

We show that the use of the spectral nested dissection (SND) ordering leads to an efficient parallel sparse Cholesky factorization on an Intel Paragon multiprocessor.

1 Introduction

A good ordering of the rows and columns of a sparse matrix can significantly reduce the storage and execution time required for factoring a sparse matrix. Good ordering algorithms currently exist for factoring sparse matrices on serial computers, but the design of ordering algorithms for the factoring of sparse matrices in parallel is an important research issue. In this paper we compare the performance of a *spectral nested dissection* (SND) algorithm, an algorithm that orders sparse symmetric positive definite matrices for efficient parallel factorization with the popular multiple-minimum-degree ordering (MMD) algorithm [4].

Nested dissection employs the divide and conquer paradigm to number symmetrically the rows and columns of a symmetric matrix from n (the order of the matrix) to one. At each step of a nested dissection ordering, a vertex separator (a set of vertices whose removal separates the residual graph into two disconnected parts) in the adjacency graph of the matrix is computed, and these vertices are numbered with the highest available numbers. Then the vertices in the two parts are numbered recursively by the same strategy. Several nested dissection ordering algorithms have been described in the literature following George's discovery of the method. The major difference between these algorithms is in the manner in which they compute a separator at each step. In earlier work we have described an algebraic algorithm that computes separators from an eigenvector of a Laplacian matrix associated with the given matrix [6]. Using this separator algorithm recursively, we have also described an algorithm for computing a spectral nested dissection ordering [7].

The essential idea in the spectral separator algorithm is to associate a symmetric Laplacian matrix $Q = D - A$ with the given symmetric matrix M, where A is the adjacency

*Department of Computer Science, Old Dominion University, Norfolk, VA 23529-0162 and ICASE, NASA Langley Research Center, Hampton, VA 23681-0001 (pothen@cs.odu.edu, pothen@icase.edu). The research of the first and fourth authors was supported by National Science Foundation grant CCR-9024954, by U. S. Department of Energy grant DE-FG02-91ER25095, and by the Canadian Natural Sciences and Engineering Research Council under grant OGP0008111 at the University of Waterloo.

†Intel Supercomputer Systems Division, 14924 NW Greenbrier Parkway, Beaverton, OR 97006 (rothberg@ssd.intel.com).

‡This author is an employee of Computer Sciences Corporation. This work was supported through NASA Contract NAS 2-12961. NASA Ames Research Center, MS T045-1, Moffett Field, CA 94035 (simon@nas.nasa.gov).

§Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802. Current Address: IBM Canada, 31/362 844 Don Mills Rd., North York, Ontario M3C 1V7 Canada (liewang@vnet.ibm.com).

matrix of M ($a_{ij} = 1$ if and only if $m_{ij} \neq 0$), and D is a diagonal matrix with d_{ii} equal to the number of off diagonal nonzeros in the ith row. An eigenvector corresponding to the smallest positive eigenvalue of Q is used to partition the vertex set into two parts, by putting all vertices with components less than or equal to the kth lowest eigenvector component into one subset, and the remaining $n - k$ vertices into the second subset. The set of edges joining the two parts is an edge separator in the graph. One choice of a vertex separator then is the smaller of the two endpoint sets (belonging to the two parts) of the edge separator. However, a maximum matching in the subgraph induced by the edge separator can be used to compute a smallest possible vertex separator from the given edge separator.

Justification for the spectral separator algorithm comes from earlier work (among others) of Fiedler, Hoffman, Mohar, Boppana, Barnes (a short survey is given in [6]), and from the recent work of Rendl and Wolkowicz [9].

Our earlier work [7] has shown that the SND ordering outperforms currently available orderings such as the Multiple-Minimum-Degree algorithm, the Sparspak Automatic Nested Dissection (AND) algorithm [1], and other variants, on several measures of parallelism by a wide margin. These results were confirmed by computing the factorization times on a an eight-processor Cray Y-MP/8.

One disadvantage of the SND ordering algorithm is that it requires a greater running time relative to the MMD ordering algorithm when the ordering is computed on a serial computer. However, many improvements have been made to the basic implementation of the spectral nested dissection algorithm to decrease its running time. An important improvement is the efficient use of vectorization in the Lanczos iteration to compute the Laplacian eigenvector. On a set of nine representative problems from the Boeing-Harwell test set, the improved SND ordering algorithm took about ten times the required by the MMD algorithm on a Cray C-90. In current work we are developing a "supernodal" SND ordering algorithm, and expect that this should lead to further decrease in the running time of the ordering algorithm. Detailed results on these improved implementations of the SND ordering algorithm will be described elsewhere [8].

In this paper, we compare the arithmetic operations, elimination tree height (this is the number of steps in a parallel factorization algorithm), and factorization time required by a panel multifrontal factorization algorithm when various ordering algorithms are employed. A longer version of this paper containing additional results is currently under preparation [5].

2 Cholesky factorization on the Intel Paragon

We compare four ordering algorithms in this section: SND1 is a spectral nested dissection algorithm in which the vertices of the adjacency graph are partitioned with respect to the median eigenvector component. In SND2, the partition is based on the sign of the eigenvector component. The former tends to compute a separator that divides the graph into two roughly equal parts at each step, while the latter tends to compute smaller separators but may divide the graph less evenly. Liu's MMD ordering algorithm [4] is currently the most popular ordering algorithm for serial computation. The fourth ordering algorithm, the MMD+JK algorithm, adapts the MMD ordering for parallel computation [2]. Here the final ordering is computed in two steps. A primary MMD ordering is used to compute a chordal filled graph (the adjacency graph of $L + L^T$) with low fill. A secondary Jess and Kees ordering then reorders the filled graph to minimize the elimination tree height

TABLE 1
Test Problems.

Problem	n	$\|A\|$	density (%)	Description
PWT	36,519	399,145	0.03	Connectivity matrix, NASA Ames pressurized wind tunnel.
FLAP	51,537	1,113,851	0.04	Part of actuator system on airplane (DAWSON5).
COPTER2	55,476	870,904	0.03	Adapted CFD grid for helicopter rotor blade, from R. Strawn, NASA Ames.
BCSSTK29	13,992	647,472	0.33	Stiffness matrix for building model of the 767 rear bulkhead.
BCSSTK33	8,738	609,380	0.80	Stiffness matrix of pin boss (auto steering component) solid elements.
BCSSTK30	28,924	4,101,340	0.49	Stiffness matrix for off-shore generator platform (MSC NASTRAN).
BCSSTK31	35,588	1,252,592	0.10	Stiffness matrix for automobile component (MSC NASTRAN).
BCSSTK32	44,609	2,103,919	0.11	Stiffness matrix for automobile chassis (MSC NASTRAN).

of the filled graph over all orderings that do not increase the fill. (This is the set of perfect elimination orderings of the chordal graph.) A composition of the two orderings is then used to factor the given matrix M. We have used a fast implementation of the Jess and Kees algorithm from [3] in combination with the best MMD ordering obtained from *twenty* runs of the MMD algorithm with randomly chosen initial orderings. This was done to decrease the elimination tree height of the MMD+JK ordering as much as possible. We excluded the Automated nested dissection (AND) algorithm from Sparspak from these experiments since it led uniformly to the slowest factorization times.

We have included three large problems obtained from NASA Ames Research Center and five of the largest structural analysis problems from the Boeing-Harwell collection in our test set (Table 1). The table shows that the latter problems are significantly denser than the former problems.

Table 2 compares the numbers of floating point operations (FLOPS) needed to compute the Cholesky factor L. The SND2 ordering leads to smaller separators than SND1 for most problems, and this is reflected in lower FLOPS as well. On five of the eight problems, the spectral orderings lead to lower FLOPS than the two minimum-degree orderings, though MMD incurs less than half the FLOPS of SND2 on the BCSSTK32 problem. In Table 3 we will see that there is a trade-off between low fill and low elimination tree heights for the denser structural analysis problems. Further, the arithmetic costs of the spectral algorithms are much less than the costs of the Automated Nested Dissection (AND) algorithm from Sparspak.

Table 3 compares elimination tree heights. The two spectral nested dissection orderings SND1 and SND2 produce much shorter elimination trees than both MMD and MMD+JK. For five out of the eight test problems, SND1 computes the shortest elimination tree, and

TABLE 2
Comparison of floating-point Operations (Millions).

Problem	SND1	SND2	MMD	MMD+JK
PWT	125	119	161	176
FLAP	818	773	1,160	1,585
COPTER2	8,715	7,505	11,378	12,472
BCSSTK29	626	512	393	395
BCSSTK33	781	882	1,204	1,137
BCSSTK30	1,865	1,600	928	823
BCSSTK31	2,623	1,919	2,551	1,992
BCSSTK32	4,097	2,526	1,109	1,018

TABLE 3
Comparison of Elimination Tree Heights.

Problem	SND1	SND2	MMD	MMD+JK
PWT	555	571	1,216	891
FLAP	1,070	1,118	1,841	1,783
COPTER2	2,597	2,477	5,863	4,058
BCSSTK29	946	923	1,758	1,197
BCSSTK33	1,254	1,344	2,414	1,740
BCSSTK30	1,392	1,649	3,644	2,748
BCSSTK31	1,723	1,750	2,362	1,970
BCSSTK32	2,023	2,014	3,173	2,381

SND does so for the remaining three. The MMD ordering leads to the longest elimination trees, which are twice as high (on the average) as the elimination trees of SND1. For all of the problems except FLAP, MMD+JK cuts down the elimination tree heights of MMD by almost 25%.

Table 4 reports the factorization times on an Intel Paragon with 64 processors. The spectral nested orderings SND1 and SND2 result in faster factorizations than MMD and MMD+JK, for all problems except for BCSSTK32. The MMD+JK ordering reduces the factorization time required by MMD by decreasing the critical path length (i.e., the longest path from a leaf to the root in the elimination tree) in the computation. On COPTER2, the largest problem in terms of serial arithmetic cost in this set, factorization with the SND2 ordering is more than twice as fast as with MMD. We attain about 950 Megaflops on this problem with the spectral ordering.

The factorization times show that although the spectral nested dissection orderings have worse FLOPs for the structural problems, which means they incur greater fill, they result in shorter and more balanced elimination trees, and lead to more efficient parallel factorization.

Consider the problem BCSSTK32 for which the SND1 ordering leads to the slowest factorization. The FLOPs SND1 requires is almost four times as many FLOPs as MMD and twice as many as SND2. Note that this is one of the denser structural analysis problems from

TABLE 4
Factorization Time(Seconds) on a 64-processor Paragon.

Problem	SND1	SND2	MMD	MMD+JK
PWT	0.46	0.42	0.75	0.67
FLAP	1.48	1.48	2.31	2.86
COPTER2	9.20	8.29	17.52	13.60
BCSSTK29	1.16	1.15	1.76	1.23
BCSSTK33	1.81	2.38	3.52	2.96
BCSSTK30	2.53	2.60	3.96	2.79
BCSSTK31	3.55	2.84	4.29	3.68
BCSSTK32	6.30	4.06	3.36	2.45

the Boeing-Harwell collection. An examination of the degree distribution of BCSSTK32 shows that most of its vertices have high degree. We suspect that this problem may not have good separators. The results show that the MMD+JK ordering may be a good parallel ordering for problems without good separators. However, for finite element meshes with good separators, the SND orderings are clearly the methods of choice among the orderings examined here.

References

[1] E. C. H. CHU, A. GEORGE, J. W. H. LIU, AND E. G. Y. NG, *User's guide for Sparspak-A: Waterloo sparse linear equations package*, Tech. Rep. CS-84-36, Computer Science, University of Waterloo, Ontario, Canada, 1989.

[2] J. JESS AND H. KEES, *A data structure for parallel L/U decomposition*, IEEE Trans. Comput., C-31 (1982), pp. 231–239.

[3] J. G. LEWIS, B. W. PEYTON, AND A. POTHEN, *A fast algorithm for reordering sparse matrices for parallel factorization*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 1146–1173.

[4] J. W. H. LIU, *Modification of the minimum degree algorithm by multiple elimination*, ACM Trans. on Math. Software, 11 (1985), pp. 141–153.

[5] A. POTHEN, E. ROTHBERG, H. D. SIMON, AND L. WANG, *Parallel sparse Cholesky factorization using spectral nested dissection ordering on the Intel Paragon.* Work in preparation, 1994.

[6] A. POTHEN, H. D. SIMON, AND K. P. LIOU, *Partitioning sparse matrices with eigenvectors of graphs*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 430–452.

[7] A. POTHEN, H. D. SIMON, AND L. WANG, *Spectral nested dissection*, Tech. Rep. CS-92-01, Computer Science, Pennsylvania State University, University Park, PA, 1992.

[8] A. POTHEN AND L. WANG, *An improved spectral nested dissection algorithm.* In preparation, 1994.

[9] F. RENDL AND H. WOLKOWICZ, *A projection technique for partitioning the nodes of a graph.* To appear in *Annals of Operation Research*, 1994.

[10] E. ROTHBERG, *Performance of panel and block approaches to sparse Cholesky factorization on the Intel iPSC/860 and Paragon multicomputers.* Proceedings of the 1994 Scalable High Performance Computing Conference (SHPCC '94), May 1994.